Image Compression Based on Fractal with Classification by Vector Quantization *

Lih-Ching Lin
Department of Applied Mathematics

Chang-Biau Yang
Kuo-Tsung Tseng
Department of Computer Science and Engineering,
National Sun Yat-sen University, Kaohsiung, Taiwan, R.O.C.
Email:cbyang@cse.nsysu.edu.tw

Abstract

The conventional fractal encoding algorithm performs an exhaustive search to find a close match between a range block and a large pool of domain blocks. For a large image, the domain pool increases obviously, so the encoding time will also increase. In this paper, we propose a hybrid scheme by combining the fractal image compression with the vector quantization. We use the longest distance first algorithm to classify the domain blocks. In this way, we can reduce the range in searching the domain pool. Experiment results show that our method can effectively speed up the encoding time about ten times. In addition, the quality of our reconstructed images is still as good as the conventional fractal algorithm.

1 Introduction

Recently, image compression becomes more and more popular with quick development of the multimedia. An image always contains a great amount of information, but some information loss is insensitive to the human eyes. Thus, we can remove such information from an image to get high compression ratio. Many image compression methods have been proposed, such as vector quantization (VQ) [5,11,15] and fractal block coding [1,4,8,9,15] and so on.

Vector Quantization (VQ) [5,11,15] is a well-known method for image compression. In VQ, we first partition the image into a set of blocks, then treat each block as a vector. For every vector, we find the closest codeword from the codebook, then use the index of the codeword to represent each vector. In the decoding phase, we find the encoded index of each vector and uses the codeword with that index to represent each vector.

The fractal image compression [1,4,6,8–10,14] utilizes the existence of local self-similarity in an image to encode the image. With this way, the fractal image compression can obtain high compression ratio and good quality of the reconstructed image.

Jacquin has pointed out the similarity between fractal image compression and VQ [8]. Both methods use a codebook to index each block, which is extracted from the original image. In the fractal image compression, the original image is partitioned into overlapping blocks as the codebook. Unlike the VQ, the fractal image compression needs not transmit the codebook to the decoder. The encoder finds a contractive operator whose fixed point is an approximation of the original image. With this contractive operator, the decoder can use any arbitrary initial image to get an approximate image by the iterative method. Therefore, the fractal image compression has very high compression ratio. However, it takes long time to encode a fractal image. The conventional fractal encoding algorithm performs an exhaustive search to find the best match from the codebook. In this paper, we use the longest distance first (LDF) algorithm to classify those overlapping blocks. With our method, we can reduce the number of blocks to be searched, thus reduce the encoding time.

In this paper, we focus on a fractal image compression with classification by vector quantization. In Section 2, we will review some related algorithms that we use in this paper. The detail of our fractal encoding

*This research work was partially supported by the National Science Council of the Republic of China under contract NSC 88-2213-E-110-012 .

algorithm is described in Section 3. The performance of our algorithm and the experiment comparison with other fractal-based algorithms are given in Section 4. Finally, we give a conclusion in Section 5.

2 Previous Works

First, we describe a simple fractal block encoding scheme [15]. The original image is partitioned into N_R nonoverlapping blocks called *range blocks*, denoted as $R_i, 1 \leq i \leq N_R$, with size $rs \times rs$. The original image is also partitioned into N_D overlapping blocks called *domain blocks*, denoted as $D_j, 1 \leq j \leq N_D$, with size $ds \times ds$. Domain block size is always larger than range block size, usually $ds = 2rs$, so we need to contract each domain block to the size of a range block. Moreover, each range block R_i finds the minimum distortion $Dist_{fractal}(t_r(D_j), R_i)$, for all D_j, and output the encoded information $I_i(flag, msg)$. A range block is said to be smooth if the variance of all pixels in the block is small. If a range block is smooth, we don't compute the distortion and use the mean of that block to represent it. With this way, we can both reduce the encoding time and increase the compression ratio.

For a given vector $Q_i = (b_1, b_2, \ldots, b_N)$, if $P = (a_1, a_2, \ldots, a_N)$ is used to represent Q_i, then the distortion between P and Q_i is defined as follows [15].

$$Dist_{fractal}(P, Q_i) = \sum_{k=1}^{N} (s_i \cdot a_k + o_i - b_k)^2, \qquad (1)$$

where s_i and o_i are defined in Equation 2 and Equation 3 respectively.

$$s_i = \frac{N \sum_{k=1}^{N} a_k b_k - (\sum_{k=1}^{N} a_k)(\sum_{k=1}^{N} b_k)}{N \sum_{k=1}^{N} {a_k}^2 - (\sum_{k=1}^{N} a_k)^2} \qquad (2)$$

$$o_i = (\sum_{k=1}^{N} b_k - s_i \sum_{k=1}^{N} a_k)/N \qquad (3)$$

$I_i(flag, msg)$ is used to represent the encoded block i, where if $flag = 0$ then $msg = \{mean\}$, otherwise, if $flag = 1$ then $msg = \{\textit{the coordinate}, r, s_i, o_i\}$.

The conventional fractal algorithm performs an exhaustive search to find a close match between a range block and a large pool of domain blocks. There are some classification schemes [4, 6, 8–10, 14] developed to reducing the number of comparisons. Here, we will introduce some of the schemes.

Jacquin [8,9] presented a scheme for classifying the domain blocks and range blocks. It is based to the classified vector quantization (CVQ) [13], and classifies the blocks into three main classes i.e. shade blocks, midrange blocks, and edge blocks. Fisher [4] also used a similar scheme with more classes.

Hamzaoui [6,14] proposed a hybrid scheme by combining fractal image compression with mean-removed shape-gain vector quantization (MRSG-VQ) [5, 12].

C. K. Lee and W. K. Lee also propose a simple method [10] based on the local varience to reduce the searching time on finding a close match between a range block and a large pool of domain blocks.

Now, we describe the algorithm that we shall use to classify the domain blocks. The longest distance first (LDF) algorithm [7] is a fast heuristic algorithm to generate better codebooks. It uses the longest distance first strategy to choose which cluster should be split instead of the maximum descent criterion in the maximum descent (MD) algorithm [2,3] and it invokes the longest distance partition technique to partition one cluster into two new clusters instead of the 2-level LBG partition technique [3] or the hyperplane partition technique [2, 3].

The distance of two vectors $P = (p_1, p_2, \cdots, p_N)$ and $Q = (q_1, q_2, \cdots, q_N)$ is defined as follows.

$$Dist_{VQ}(P, Q) = \sum_{j=1}^{N} (p_j - q_j)^2 \qquad (4)$$

The longest distance partition algorithm is as follows:

Algorithm Longest Distance Partition (LDP)

Input: The splitting cluster $C_i = \{x_1, x_2, \ldots, x_{n_i}\}$.

Output: Two new clusters C_a and C_b and their representative codewords.

Step 1: Calculate the centroid v_i of the splitting cluster C_i.

Step 2: Find x_a such that $Dist_{VQ}(x_a, v_i) = \max_{1 \leq j \leq n_i} Dist_{VQ}(x_j, v_i)$.

Step 3: Find x_b such that $Dist_{VQ}(x_b, x_a) = \max_{1 \leq j \leq n_i} Dist_{VQ}(x_j, x_a)$.

Step 4: Split cluster C_i into C_a and C_b. That is, if $Dist_{VQ}(x_j, x_a) < Dist_{VQ}(x_j, x_b)$ then $x_j \in C_a$; otherwise $x_j \in C_b$, $1 \leq j \leq n_i$.

Step 5: Calculate the centroids of C_a and C_b as the two new codewords.

In order to reduce the partition time, the LDP algorithm uses a fast method [3] to determine which cluster x should belong to. The fast method [3] is as follows:
Let $x_j = (\alpha_1, \alpha_2, \ldots, \alpha_N)$, $x_a = (a_1, a_2, \ldots, a_N)$ and $x_b = (b_1, b_2, \ldots, b_N)$.
Then x_j is put in C_a if

$$\sum_{j=1}^{N}(\alpha_j - a_j)^2 < \sum_{j=1}^{N}(\alpha_j - b_j)^2. \quad (5)$$

So, x is placed in C_a if

$$\sum_{j=1}^{N}(a_j - b_j)\alpha_j > \frac{1}{2}\sum_{j=1}^{N}(a_j^2 - b_j^2). \quad (6)$$

The values of $(a_j - b_j)$ and $\frac{1}{2}\sum_{j=1}^{N}(a_j^2 - b_j^2)$ in Equation 6 are not changed during the splitting process and can be pre-calculated. The amount of computation can be reduced to N multiplications, $N-1$ additions and 1 comparison. So Equation 6 is used instead of Equation 5.

Now, we will describe the longest distance first algorithm. The longest distance first algorithm chooses which cluster should be split. It finds the cluster with the maximum longest distance and applies the LDP algorithm to split this cluster.

The longest distance first algorithm is as follows:

Algorithm Longest Distance First (LDF)

Input: The codebook size and the training set.

Output: The codebook.

Step 1: Split the entire training set into two new clusters by the LDP algorithm.

Step 2: Let the two newly formed clusters be C_a and C_b. Find the longest distances in C_a and C_b respectively.

Step 3: Select the cluster with the maximum longest distance in all clusters. Split the selected cluster into two new clusters by using the LDP algorithm.

Step 4: If the number of current clusters is equal to the codebook size we desire, then output the centroids of the clusters as the codebook and stop; otherwise go to Step 2.

The advantages of the LDF algorithm are its speed and quality. It requires much less time than other codebook generation algorithms. Moreover, the quality of the codebooks generated by LDF is very good, so we choose the LDF algorithm as a base on our fractal algorithm.

3 The Fractal Encoding with VQ Classification

The conventional fractal algorithm spends too much time on finding the best match between a range block and a large pool of domain blocks. In order to reduce the encoding time, we decrease the search pool by clustering all domain blocks. Our clustering algorithm is based on vector quantization (VQ).

For training a local codebook in VQ, all training vectors are partitioned from an original image. In the end of classification, similar vectors will be put into the same cluster. Thus we utilize this concept to classify the domain blocks. Besides, an efficient codebook generation algorithm for VQ is very important. To generate the codebook, we choose an efficient method, the longest distance first (LDF) algorithm which we have introduced in the previous section.

We can classify the domain blocks efficiently by using the LDF algorithm. First, we extract N_D overlapping domain blocks from the original image and contract each domain block to the size of a range block. The elements of the training set are the domain blocks after applied the eight transformations. Thus, the size of the training set is $8N_D$. After applying the LDF algorithm, we get N_C codewords and classify each transformed domain block into a correlative cluster. Each range block finds a nearest codeword (cluster) in the codebook and then in the cluster, finds the transformed domain block with minimum distortion to the range block. Finally, the encoded information is output.

Our fractal encoding algorithm is as follows:

Basic Phase

Input: An original image.

Output: The previous processing results.

Step 1: Partition the original image into N_R nonoverlapping range blocks, denoted as $\mathcal{R}=\{R_1, R_2, \ldots, R_{N_R}\}$.

Step 2: Extract N_D overlapping domain blocks from the original image, denoted as $\mathcal{D}=\{D_1, D_2, \ldots, D_{N_D}\}$.

Step 3: Contract each domain block to the size of a range block.

Step 4: For each domain block D_j, calculate it's variance σ_{D_j}. If $\sigma_{D_j} < T_\sigma$, where T_σ is a predefined threshold, then remove D_j from $\mathcal{D}$.

Step 5: Use the LDF algorithm to split all $t_r(D_j)$, $1 \leq r \leq 8$, until the number of clusters achieves N_C, where N_C is a predefined codebook size. The set of all clusters is denoted as $\mathcal{C}=\{C_1, C_2, \ldots, C_{N_C}\}$ and the codeword of each cluster C_k is denoted as CW_k, $1 \leq k \leq N_C$.

Algorithm A

Input: An original image.

Output: The encoding information.

Steps 1-5: Basic Phase.

Step 6: For each range block R_i, calculate its mean $\overline{R_i}$ and variance σ_{R_i}. If $\sigma_{R_i} < T_\sigma$, then output $I_i(0, \overline{R_i})$; otherwise do Steps 7-8.

Step 7: Find k such that $Dist_{fractal}(CW_k, R_i)$ is the minimum, where $1 \leq k \leq N_C$.

Step 8:
Find $t_r(D_j)$ such that $Dist_{fractal}(t_r(D_j), R_i)$ is the minimum, $\forall t_r(D_j) \in C_k$. Then output $I_i(1, \textit{the coordinate of} D_j, r, s_i, o_i)$.

In Algorithm A, if a codebook of large size is built, it needs more time in Step 5 and Step 7. However, the search time can be reduced in Step 8. In addition, the quality of the reconstructed image increases very little with a large codebook, so we generate a codebook with a median size. The time required for algorithm A is little, but we find that the quality of the reconstructed image is not good enough. Thus, we modify algorithm A to algorithm B by increasing the search window on the codebook. That is, when the close match $t_r(D_j)$ is found, more than one cluster is searched.

Algorithm B

Input: An original image.

Output: The encoding information.

Steps 1-5: Basic Phase.

Step 6: Define a search window size, W.

Step 7: For each range block R_i, calculate its mean $\overline{R_i}$ and variance σ_{R_i}. If $\sigma_{R_i} < T_\sigma$, then output $I_i(0, \overline{R_i})$; otherwise do Steps 8-9.

Step 8: Find a set $S = \{s_1, \cdots, s_W\}$ such that $Dist_{fractal}(CW_k, R_i) \geq Dist_{fractal}(CW_{s_m}, R_i)$, $\forall k \notin S$ and $\forall s_m \in S$.

Step 9: Find
$t_r(D_j)$ such that $Dist_{fractal}(t_r(D_j), R_i)$ is the minimum, $\forall t_r(D_j) \in C_{s_m}$, $m = 1, \cdots, W$. Then output $I_i(1, \textit{the coordinate of} D_j, r, s_i, o_i)$.

In order to obtain better quality, we give a search window size W for searching more clusters in algorithm B. Experiments also show that large window size will obtain the reconstructed images with better quality, but the encoding time increases too. Thus, we again modify the algorithm by adding a threshold T. We use T to determine if a transformed domain block $t_r(D_j)$ is good enough to represent the range block. If it is, we do not search other clusters furthermore.

Algorithm Longest Distance First Fractal Encoding

Input: An original image.

Output: The encoding information.

Steps 1-5: Basic Phase.

Step 6: Define a search window size, W, and a threshold, T.

Step 7: For each range block R_i, calculate its mean $\overline{R_i}$ and variance σ_{R_i}. If $\sigma_{R_i} < T_\sigma$, then output $I_i(0, \overline{R_i})$; otherwise do Steps 8-9.

Step 8: Find a set $S = \{s_1, \cdots, s_W\}$ such that $Dist_{fractal}(CW_k, R_i) \geq Dist_{fractal}(CW_{s_m}, R_i)$, $\forall k \notin S$ and $\forall s_m \in S$.

Step 9:
Find $t_r(D_j)$ such that $Dist_{fractal}(t_r(D_j), R_i) < T$ or $Dist_{fractal}(t_r(D_j), R_i)$ is the minimum, $\forall t_r(D_j) \in C_{s_m}$, $m = 1, \cdots, W$. Then output $I_i(1, \textit{the coordinate of } D_j, r, s_i, o_i)$.

The longest distance first (LDF) fractal encoding algorithm effectively reduces the encoding time with the threshold T. Small threshold T will obtain the reconstructed images with better quality, but the reduced time is not as much as that with large threshold. We use the LDF fractal encoding algorithm to compare with other fractal encoding algorithms in this paper. Our experiment results are listed in the next section.

4 Experiment Results and Performance Analysis

In this section, we show our experiment results and analyze the performance of our algorithms. Our algorithm is implemented by Borland C++ Builder on PC with Intel Celeron™ processor 300A MHz and 64 MB RAM. Our testing images include "Lena", "F16", "Pepper" and "Baboon". All of these images are of 256 gray levels with resolution 256×256.

To measure the quality of the reconstructed image, we use the peak signal-to-noise ratio (PSNR), which is defined as:

$$\mathrm{PSNR} = 10 \log_{10}\left[\frac{255^2}{\frac{1}{L\times L}\sum_{i=1}^{L}\sum_{j=1}^{L}(x_{ij}-\hat{x}_{ij})^2}\right],$$

where $L \times L$ = size of image, x_{ij} = pixel value of the original image at coordinate (i,j), and $\hat{x}_{ij}$ = pixel value of the reconstructed image at coordinate (i,j) [3,7].

All decoding process in this paper uses an image with the initial value of each pixel is 128. And s_i and o_i in Equation 2 and Equation 3 are quantized to 3 bits and 7 bits respectively.

Now, we would like to show some of our experiment results. Table 1 shows the PSNR and the encoding time of our algorithm with various parameters. We find that large search window size will get better quality. However, a small search window size will reduce the encoding time. According to the expriment results, we can get near PSNR if the threshold $T = 9 \times N$. Thus, the best parameters in our algorithm are that codebook size = 500, window size = 15 and $T = 9 \times N$.

Table 2 shows the comparison of the PSNRs in iterations 1 through 9 for the conventional fractal algorithm, the local variance fractal algorithm and our algorithm. The PSNR of our algorithm converges after the sixth iteration but the local variance algorithm converges after the 8th or 9th iteration. Finally, the performance analysis is summarized in Table 3. The relative speedup of our algorithm to the conventional exhaustive search algorithm is about ten times. Not only our algorithm is faster than conventional fractal algorithm, but also the quality of our reconstructed images is as good as that of the conventional fractal algorithm. Our algorithm is also faster than the local variance algorithm. We also test for the 512×512 Lena image and the experiment results are shown in Table 4. In Table 4, we find that the performance of our algorithm with the 512×512 Lena image is also very good.

Table 1: The PSNR and time of our algorithm with various parameters. N_C: codebook size, W: search window size, T: threshold, N: range block size $\times$ range block size. Test image is Lena 256×256. (a) Domain block size: 16×16; range block size: 8×8; bpp: 0.379. (b) Domain block size: 8×8; range block size: 4×4; bpp: 1.232.

Parameters $N_C = 500$	PSNR	Time (sec) LDF clustering	Time (sec) Fractal encoding	Time (sec) Total
$W = 1$	27.4351	197.735	52.212	249.947
$W = 5$	27.9438	197.870	179.116	376.986
$W = 10$	28.0895	197.824	337.957	535.781
$W = 15$	28.1305	197.850	473.223	671.073
$W = 15$ $T = 9N$	28.1149	198.784	302.135	500.919
$W = 15$ $T = 16N$	28.0427	198.815	253.136	451.951

(a)

Parameters $N_C = 500$	PSNR	Time (sec) LDF clustering	Time (sec) Fractal encoding	Time (sec) Total
$W = 1$	32.8442	93.449	115.315	208.764
$W = 5$	33.5797	93.735	376.951	470.686
$W = 10$	33.7773	93.760	723.731	817.491
$W = 15$	33.8678	93.751	1004.328	1098.079
$W = 15$, $T = 9N$	33.6956	86.066	407.394	493.460

(b)

5 Conclusion

In this paper, we propose a fast encoding algorithm for fractal image compression based on vector quantization. In the conventional fractal encoding algorithm, each range block performs an exhaustive search to find a best match from the domain pool, so a large domain pool will significantly increase the search time. Thus, we propose a scheme to classify the domain blocks by using the longest distance first (LDF) algorithm. In average, our method can reduce the search space to $\frac{\textit{search window size}}{\textit{codebook size}} = \frac{W}{N_C}$ at least, the percentage is $\frac{15}{500} = 3\%$ in this paper. Theoretically, we can also reduce the same percent of the encoding time. That is, we should reduce the encoding time about 33 times when the percentage is 3%, but actually it is only about ten times. The reason is, the overhead in training the codebook and finding the closest codeword set. Thus, how to decrease the overhead is one of our future works.

Experiment results show that our method is faster than the conventional fractal encoding method and the local variance method, and we still have good quality of the reconstructed images under the same compression ratio.

References

[1] M. F. Barnsley, *Fractals everywhere*. San Diego, USA: Academic Press, Inc., 1988.

Table 2: Comparison of the PSNRs after executing 1 through 9 iterations. (a) Domain block size: 16×16; range block size: 8×8; bpp: 0.379. (b) Domain block size: 8×8; range block size: 4×4; bpp: 1.232.

# of iterations	PSNR		
	Conventional fractal	LDF fractal	Local variance fractal
1	19.135	19.416	18.077
2	22.594	22.914	21.260
3	25.470	25.737	24.118
4	27.319	27.444	26.155
5	27.982	28.044	27.140
6	28.145	28.167	27.396
7	28.153	28.144	27.464
8	28.142	28.125	27.478
9	28.136	28.115	27.479

(a)

# of iterations	PSNR		
	Conventional fractal	LDF fractal	Local variance fractal
1	21.031	21.710	20.010
2	25.769	26.457	24.254
3	29.989	30.482	28.316
4	32.680	32.757	31.397
5	33.687	33.514	32.903
6	33.950	33.711	33.421
7	33.982	33.710	33.541
8	33.967	33.702	33.559
9	33.959	33.696	33.551

(b)

Table 3: Performance of various fractal algorithms. LDF fractal is our algorithm; the window size of local variance fractal I is 3000. The window size of local variance fractal II in (a) is 10000 and in (b)is 24000. (a) Domain block size: 16×16; range block size: 8×8; (b) Domain block size: 8×8; range block size: 4×4.

Algorithm		Conventional fractal	LDF fractal	Local variance fractal I	Local variance fractal II
Lena bpp= 0.379	PSNR	28.136	28.115	27.159	27.479
	Time (sec)	5262.174	500.919	541.105	895.134
F16 bpp= 0.339	PSNR	25.792	25.630	25.113	25.293
	Time (sec)	4369.355	425.703	451.424	746.070
Pepper bpp= 0.461	PSNR	26.957	26.884	26.211	26.484
	Time (sec)	5722.120	610.994	586.935	972.130
Baboon bpp= 0.400	PSNR	23.149	23.075	22.801	22.918
	Time (sec)	7073.116	637.094	717.960	1173.390

(a)

Algorithm		Conventional fractal	LDF fractal	Local variance fractal I	Local variance fractal II
Lena bpp= 0.379	PSNR	33.959	33.696	32.331	33.551
	Time (sec)	5345.645	493.460	536.320	2081.676
F16 bpp= 1.176	PSNR	32.195	32.025	30.290	32.000
	Time (sec)	4895.065	552.324	500.595	1927.050
Pepper bpp= 1.759	PSNR	33.341	33.147	31.964	32.912
	Time (sec)	6032.285	616.644	589.800	2277.370
Baboon bpp= 1.316	PSNR	27.106	26.952	25.388	26.514
	Time (sec)	9580.355	1200.984	980.895	3636.090

(b)

[2] C. K. Chan and C. K. Ma, "Maximum descent method for image vector quantization," *Electronics Letters*, Vol. 27, No. 19, pp. 1772–1773, Sep. 1991.

[3] C. K. Chan and C. K. Ma, "A fast method of design better codebooks for image vector quantization," *IEEE Transactions on Communications*, Vol. 42, No. 2/3/4, pp. 237–243, Feb./Mar./Apr. 1994.

[4] Y. Fisher, *Fractal image compression: theory and application*. New York, USA: Springer-Verlag, 1994.

[5] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Boston, USA: Kluwer Academic Publishers, second ed., 1992.

[6] R. Hamzaoui, "Codebook clustering by self-organizing maps for fractal image compression," *Fractals*, Vol. 2, No. 0, 1994.

[7] M. C. Huang and C. B. Yang, "Fast algorithm for designing better codebooks in image vector quantization," *Optical Engineering*, Vol. 36, pp. 3265–3271, Dec. 1997.

[8] A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Transactions on Image Processing*, Vol. 1, No. 1, pp. 18–30, Jan. 1992.

[9] A. E. Jacquin, "Fractal image coding: a review," *Proceedings of the IEEE*, Vol. 81, No. 10, pp. 1451–1465, Oct. 1993.

[10] C. K. Lee and W. K. Lee, "Fast fractal image block coding based on local variances," *IEEE Transactions on Image Processing*, Vol. 7, No. 6, pp. 888–891, June 1998.

[11] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, Vol. C-28, No. 1, pp. 84–95, Jan. 1980.

[12] T. Murakami, K. Asai, and E. Yamazaki, "Vector quantization of video signals," *Electronics Letters*, Vol. 7, pp. 1005–1006, 1982.

[13] B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Transactions on Communications*, Vol. 34, No. 11, pp. 1105–1115, Nov. 1986.

[14] D. S. Raouf Hamzaoui, Martin Müller, "VQ-Enhanced fractal image compression," *IEEE International Conference on Image Processing (ICIP'96), Lausanne, Sept. 1996*, pp. 1–4.

Table 4: Performance of various fractal algorithms with the 512×512 Lena image. the parameters in our algorithm are that codebook size = 2000, window size = 15 and $T = 9 \times N$. (a) Domain block size: 16×16; range block size: 8×8; bpp: 0.380; the window size of local variance fractal is 40000. (b) Domain block size: 8×8; range block size: 4×4; bpp: 1.289; the window size of local variance fractal is 96000.

Algorithm	(a)		(b)	
	PSNR	Time (sec)	PSNR	Time (sec)
Conventional fractal	29.990	131968.480	35.058	88763.863
LDF fractal	29.860	20665.195	34.817	7297.361
Local variance fractal	29.379	34826.097	34.587	31865.435

[15] S. C. Tai, *Data compression*. Taipei, Taiwan: Unalis, second ed., 1998.